Class 4 - Data Binding

Resumo:

- Detailed example of Data Binding
- Data Binding to an XML file
- Validation with WPF

This manual assumes that the resolution of the scripts of the last two classes was successfully implemented. If you have not completed Class 3, you can start in section 4.2 using the code for Class 2 up to point 2.5 (a possible solution is available on the course website).

4.1. DataGrid

Replace in the interface (if you are using the example provided) the two labels "Disciplina" and "ECTS" with a `DataGrid`. To create the `DataGrid` you can use the following code:

```
<DataGrid Grid.Column="1" Grid.Row="2" >
        <DataGrid.Columns>
            <DataGridTextColumn Header="Disciplina"/>
            <DataGridTextColumn Header="ECTS"/>
        </DataGrid.Columns>
</DataGrid>
```

Apply the columnHeaderStyle style to the Datagrid (associate the style with the ColumnHeaderStyle attribute and not the Style) by using the following code:

```
<!-- DataGrid header style -->
  <Style x:Key="columnHeaderStyle" TargetType="{x:Type
DataGridColumnHeader}">
   <Setter Property="Height" Value="35" />
       <Setter Property="Padding" Value="5" />
       <Setter Property="Background" Value="#4E87D4" />
       <Setter Property="Foreground" Value="White" />
  </Style>
```

4.2. Data Binding to Data in XML Format

On the `DetiHome.xaml` page, add the following code to the `<Grid>` element that defines the information associated with each course. In this case an `XmlDataProvider` is used to provide information that could be loaded from an XML file or from a Database. If you are using data binding from the last class, remove the associated code as the binding will now be made to XML data.

```
<Grid.Resources>
<!-- courses 1º year and 1º semester-->
        <XmlDataProvider x:Key="FonteDadosDeti" XPath="Cursos">
                    <x:XData>
                   <Cursos xmlns="">
                       <Curso Nome="Computadores e Telemática" Codigo="8240">
                           <UC Disciplina="Álgebra Linear e Geometria Analítica" ECTS="6" />
                           <UC Disciplina="Cálculo I" ECTS="6" />
                           <UC Disciplina="Programação I" ECTS="8" />
                           <UC Disciplina="Introdução aos Sistemas Digitais" ECTS="6" />
                       </Curso>
                       <Curso Nome="Electrónica e Telecomunicações" Codigo ="8204">
                           <UC Disciplina="Álgebra Linear e Geometria Analítica" ECTS="6" />
                           <UC Disciplina="Cálculo I" ECTS="6" />
                           <UC Disciplina="Programação I" ECTS="8" />
                           <UC Disciplina="Introdução aos Sistemas Digitais" ECTS="6" />
                       </Curso>
                       <Curso Nome="Engenharia Informática" Codigo="8295">
                           <UC Disciplina="Álgebra Linear e Geometria Analítica" ECTS="6" />
                           <UC Disciplina="Cálculo I" ECTS="6" />
                           <UC Disciplina="Elementos de Física" ECTS="8" />
                           <UC Disciplina="Introdução às Tecnologias Web" ECTS="6" />
                           <UC Disciplina="Fundamentos de programação" ECTS="6" />
                       </Curso>
                   </Cursos>
               </x:XData>
        </XmlDataProvider>
</Grid.Resources>
```

Use a `Data Template` to indicate which information should be used to map in the `ListBox`. You can define this `Data Template` also in the section `Grid Resources`.

```
        <DataTemplate x:Key="nameItemTemplate">
                <Label Content="{Binding XPath=@Nome}"/>
      </DataTemplate>
```

Modify the `ListBox` in the DETI-Home page to take into account the data provided through the following code that indicates the data source to be used (`ItemSource`) and the format to be applied to the data (`ItemTemplate`).

```
        <ListBox Name="cursosListBox" Grid.Column="1" Grid.Row="2"
        ItemsSource="{Binding Source={StaticResource FonteDadosDeti}, XPath=Curso}"
          ItemTemplate="{StaticResource nameItemTemplate}">
        </ListBox>
```

This code defines the data source (`FonteDadosDeti`) and uses a (`nameItemTemplate`) to ensure that the name is displayed for all available elements (courses).

4.3. Association Between Data and Controls (Data Binding)

It is now necessary to use the information to map the correct Disciplinas for each course. For this, it is necessary to transfer the data to the DETI-Cursos page. This can be done through the constructor and using the following code that indicates that the `DataContext` of the page contains all the data defined in the `Resource`.

Note that if you have any code associated with the click on the button, you must remove it.

```
// Constructor of the class Deti-Cursos
public DETI_Cursos(object data) : this()
{
    // Binds the data to the context of the new page.
    this.DataContext = data;
}
```

The transfer of data to this page can be done now by clicking on the "Ver" button with the following code:

```
//See page Disciplinas
    DETI_Cursos PaginaCursos= new DETI_Cursos
(this.cursosListBox.SelectedItem);
    this.NavigationService.Navigate(PaginaCursos);
```

It's now necessary to map the data and the controls on the Cursos page. Start by associating the name of the selected course with the label Disciplina through Data Binding:

```
        <Label Style="{StaticResource labelStyle}" Content="{Binding
XPath=@Nome}"></Label>
```

Finally, you can associate the information of the Disciplinas and ECTS with the DataGrid. Note that if you do not set the `AutoGenerate` option to `false`, the columns are automatically generated according to the data source.

```
<DataGrid ItemsSource="{Binding XPath=UC}" AutoGenerateColumns="False" >
    <DataGrid.Columns>
      <DataGridTextColumn Header="Disciplina" Binding="{Binding
      XPath=@Disciplina}"/>
      <DataGridTextColumn Header="ECTS" Binding="{Binding XPath=@ECTS}" />
      </DataGrid.Columns>
</DataGrid>
```

More information about binding in WPF can be found at:
http://msdn.microsoft.com/en-us/library/ms752347(v=vs.110).aspx
http://msdn.microsoft.com/en-us/library/aa480224.aspx

4.4. Reading na XML File

Modify the code to read information directly from an xml file (use for example the deti.xml file). To do this, modify the `xmlDataProvider` to use the following code:

```
<XmlDataProvider x:Key="FonteDadosDeti"
 Source="Deti.xml" XPath="Cursos"/>
```

Add the Deti.xml file to the project and run the program. Modify some parameters of the file and see the result.

Modify the `DataTemplate nameItemTemplate` to show the code of the course followed by its name in the `ListBox`. You can add a `StackPanel` with horizontal orientation by placing several `TextBlock` with the text associated with the correct binding. Use a similar approach to add the code in the discipline label on the DETI-Cursos page.

4.5. Validation in WPF

Add, in a position to your choice of the grid on the DETI_Home page, a textbox to specify the course number to select. We will add the code to validate the content ensuring that the textbox has the code for one of the 3 courses.

Start by adding the validation function to the code by inheriting the ValidationRule class. You can use the following code:

```
public class NameValidator : ValidationRule
        {
            public override ValidationResult Validate (object value,
System.Globalization.CultureInfo cultureInfo)
            {
                if (value.ToString() != "8240" && value.ToString() != "8204" &&
value.ToString() != "8295")
                    return new ValidationResult(false, "Tem que usar um código
válido");

                return ValidationResult.ValidResult;
            }
```

Add a style and a controlTemplate to the resources of the xaml grid to indicate the occurrence of an error, for example using the following code:

```
            <Style x:Key="textBoxInError" TargetType="{x:Type TextBox}">
              <Style.Triggers>
                  <Trigger Property="Validation.HasError" Value="true">
                      <Setter Property="ToolTip"
       Value="{Binding RelativeSource={x:Static RelativeSource.Self},
                      Path=(Validation.Errors)[0].ErrorContent}"/>
                  </Trigger>
              </Style.Triggers>
```

```
        </Style>

    <ControlTemplate x:Key="validationTemplate">
        <DockPanel>
            <TextBlock Foreground="Red" FontSize="20">!</TextBlock>
            <AdornedElementPlaceholder/>
        </DockPanel>
    </ControlTemplate>
```

It is now possible to bind the textbox to the validation rule with the following code in the textbox:

```
      <TextBox Width="100"  HorizontalAlignment="Left"
            Validation.ErrorTemplate="{StaticResource validationTemplate}"
             Style="{StaticResource textBoxInError}">
        <TextBox.Text>
            <Binding Source="{StaticResource myCurso}" Path="CodCurso"
UpdateSourceTrigger="PropertyChanged">
            <Binding.ValidationRules>
                    <local:NameValidator/>
            </Binding.ValidationRules>
            </Binding>
        </TextBox.Text>
      </TextBox>
```

Note that the binding is made to the course code of the Course class defined in the last class. To use this code you must instantiate the course class in xaml using:

```
<local:Curso x:Key="myCurso"/>
```